Programación orientada a objetos

Nombre: Nicolas Aldair Gualoto Pillajo

NRC: 1323

MGTR: Luis Enrique Jaramillo Montaño

Actividad 1

¿Qué es el paradigma de la programación orientada a objetos?

La programación orientada a objetos (POO) es un paradigma de programación, un modelo o estilo de programación que proporciona guías sobre como trabajar con el basado en clases y objetos. Este tipo de programación permite estructurar un programa en piezas simples y reutilizables (clases) para crear instancias individuales de objetos.

Con el paradigma de POO lo que busca es dejar de centrarse en la lógica pura de los programas para comenzar a pensar en objetos, lo que forma la base de dicho paradigma. Esto ayuda bastante en sistemas grandes, pues en lugar de pensar en funciones, se piensa en las relaciones o interacciones de los distintos elementos del sistema.

Principios de la POO

1- Encapsulación: Presenta toda la información importante de un objeto dentro del mismo y solo expone la información elegida al mundo exterior. Agrupa características con acceso privado y comportamientos con acceso público.

Ejemplo:

Para explicar la encapsulación usaremos un coche de ejemplo

El coche comparte información publica mediante las luces de freno o intermitentes para indicar los giros (interfaz pública). Por contra, la interfaz interna, el mecanismo propulsor del coche, está oculto bajo el capó. Al conducir un automóvil es necesario indicar a otros conductores

los movimientos, pero no exponer datos privados sobre el tipo de combustible o la temperatura del motor, ya que son muchos datos lo que confundiría a los demás conductores. (oculta su mecanismo I.)

2.- Abstracción

La abstracción se produce cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto. Facilita el mantenimiento de programas grandes, donde los objetos se comunican entre sí, ocultando detalles complejos al usuario. Es una extensión de la encapsulación.

Continuando con el ejemplo anterior, no es necesario que conozcas todos los detalles sobre cómo funciona el motor de un coche para poder conducirlo

3.- La herencia.

Define relaciones jerárquicas entre clases, de modo que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y métodos, permitiendo la ejecución de distintos comportamientos a las clases secundarias. Mediante la definición en una clase de los atributos y comportamientos básicos, pueden crearse clases secundarias, ampliando la funcionalidad de la clase principal y añadiendo atributos y comportamientos extra. Es una de las claves de la POO.

Podemos usar como ejemplo a los animales, pueden emplearse una única clase de animal y añadir un atributo de tipo animal que especifique el tipo de animal. Los distintos tipos de animales requerirán diferentes métodos, por ejemplo, los reptiles deben poner huevos y los peces nadar. Incluso si los animales disponen de un método en común, como moverse, la implementación requeriría muchas declaraciones "si" para garantizar el comportamiento de movimiento idóneo. Por ejemplo, las ranas saltan, mientras que las serpientes se deslizan. El principio de herencia permite solucionar dicho problema.

4- Polimorfismo

Reside en diseñar objetos para compartir comportamientos, lo que permite procesar objetos de distintos modos. Es la capacidad de presentar la misma interfaz para distintas maneras subyacentes o tipos de datos. Al usar la herencia, los objetos pueden anular los comportamientos principales compartidos, con comportamientos secundarios específicos. El polimorfismo permite que el mismo método ejecute distintos comportamientos de dos modos: anulación de método y sobrecarga de método.

Beneficios de Programación Orientada a Objetos.

- Reutilización del código
- Convierte cosas complejas en estructuras simples reproducible
- Evita la duplicación de código
- Permite trabajar en equipo gracias al encapsulamiento, puesto que minimiza la posibilidad de duplicar funciones cuando distintas personas trabajan sobre un mismo objeto al mismo tiempo
- Al estar la clase bien estructurada permite la corrección de errores en diversos lugares del código
- Protege la información mediante la encapsulación, pues solo se puede acceder a los datos del objeto mediante propiedades y métodos privados.
- La abstracción nos permite construir sistemas más complejos y de un modo más sencillo y organizado

¿Qué es una clase, un objeto, un atributo y un método?

En programación, una clase, un objeto, un atributo y un método son conceptos fundamentales que nos permiten modelar entidades y su comportamiento, tambien para organizar y estructurar el codigo de manera eficiente.

~Clase: Es una plantilla que define las características de los objetos que se crean a partir de ella, incluye atributos (propiedades) y métodos (funcionalidades) que los objetos tendrán.

~Objeto: Es una instancia de una clase específica, representa un elemento real o conceptual que puede ser manipulado dentro del programa. Con esta explicación podemos deducir que cada objeto tiene su propio conjunto de datos, que son valores específicos de los atributos definidos en su clase, y puede ejecutar las acciones establecidas por los métodos.

~Atributo: Los atributos son las propiedades o características que describen el estado de un objeto. Estos atributos se definen en la clase y cada objeto puede tener valores específicos para ellos, permitiendo que los objetos sean únicos incluso si estos pertenecieran a la misma clase.

~Método: Basicamente sería una Función definida dentro de una clase que representa el comportamiento de un objeto.

¿Qué es un sistema de control de versionamiento y para que sirve?

Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo.

¿Para qué sirve?

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código es un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Protección del código fuente: Actúa como una capa de seguridad para proteger el código, que es una

de los activos más valiosos de un proyecto de software, frente a errores humanos, catástrofes o pérdida accidental.

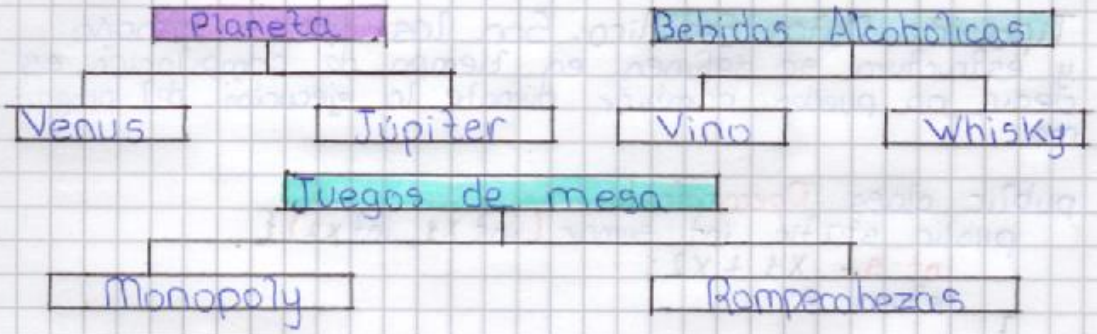
Colaboración eficiente: Permite que múltiples desarrollladores trabajen simultaneamente en diferentes partes del proyecto sin interferir entre sí, incluso en el mismo archivo, gracias a la gestión de ramas y fusiones.

Resolución de conflictos: Detecta y facilita resolución de conflictos cuando dos desarrolladores realizan cambios incompatibles en la misma parte del código.

Historial completo del proyecto: Mantiene un registro detallado de todos los cambios realizados en el proyecto a lo largo del tiempo.

Soporte multiplataforma: Algunos SCV, como Git, funcionan en cualquier sistema operativo, permitiendo que los equipos trabajen con sus herramientas preferidas sin restricciones.

Realizar tres UML de dos clases hijos una padre.



Dato extra: Un UML es un lenguaje visual estandarizado que se usa en ingenierías de software para diseñar, entender y documentar sistemas. Sirve para representar tanto la estructura estática (diagramas de clases, objetos, componentes) como el comportamiento dinámico (diagramas de casos de uso, secuencia, actividades)

Consulta

Tipos de Datos Primitivos y Referenciados

Datos Primitivos: Son los tipos de datos básicos que ya están integrados en Java, almacenan directamente el valor.

El lenguaje JAVA da de base una serie de tipos de datos primitivos.

- byte
- short
- int
- long
- float
- double
- boolean
- char

```
public class Main {
    public static void main (String[]args {
        String saludo= "Hola mundo";
        System.out.println (saludo);
    }
}
```

Datos Referenciados: Se refieren a los objetos que se almacenan en la memoria y cuya referencia se guarda en una variable. (En lugar de almacenar directamente el valor de los datos, como sucede con los datos primitivos)

Tipo de Datos Estático. Son los que su tamaño y estructura se definen en tiempo de compilación, es decir, no pueden cambiar durante la ejecución del programa.

```
public class Operación {
    public static int sumar (int x1, int x2) {
        int s= x1 + x2;
        return s;
    }

    public static int restar (int x1, int x2) {
        int r = x1 - x2;
        return r;
    }
{
```

Tipo de Datos Dinámicos: Son aquellos que permiten cambiar su tamaño o estructura en tiempo de ejecución. Esto contrasta con los tipos de datos estáticos, cuyo tamaño y estructura están definidos en el momento de la compilación.

- ArrayList
- LinkedList

```
public class RecorrerListaForEach {
    public static void main (String [] args ){
        ArrayList<String> lista = new ArrayList<>();

        lista.add ("Manzana");
        lista.add ("Durazno");
        lista.add ("Tomate");

        System.out.println ("Lista de frutas");
        for (String fruta : lista) {
            System.out.println(Fruta);
        }
    }
}
```